Amendments to the Claims

Listing of Claims:

Claims 1-16 (cancelled)

Claim 17. (currently amended) A Java run-time system comprising:

a stack-based interpreter for executing a Java program comprising Java bytecode instructions and Java class structures;

a converter for mapping standard Java symbolic linking strings contained in a downloaded Java program onto linking identifiers, wherein the converter is adapted to use a hash function to map the standard Java symbolic linking strings onto linking identifiers; and

an export table for storing linking identifiers generated by the converter to bind a reference in a bytecode instruction to be executed to a corresponding link target;

wherein the converter is adapted to use a parameterized hash function to map the standard Java symbolic linking strings onto linking identifiers, a parameter for the hash function being contained in the Java program to be downloaded, so that no additional linking information is necessary to translate standard Java class files to Java cap files.

Claim 18. (cancelled)

Claim 19. (currently amended) The Java run-time system of claim 17, wherein the parameter is used so that different symbolic linking strings are not mapped to the same linking ~~string~~ identifiers.

Claim 20. (currently amended) The Java run-time system of claim 17, wherein each linking ~~string~~ identifier is a short token.

Claim 21. (previously presented) The system of claim 20 wherein the short token is a number.

Claim 22. (previously presented) The system of claim 20 wherein the short token is a short integer.

Claim 23. (previously presented) The Java run-time system of claim 17, wherein the run-time system is ported on an embedded microcontroller of a smart card.

Claim 24 (currently amended). A method for downloading and linking a Java program on a Java run-time system comprising a stack-based interpreter for executing bytecode instructions, the Java program comprising Java bytecode instructions and Java class structures and including Java standard symbolic linking strings to bind a reference in a bytecode instruction to be executed to a corresponding link target, the method comprising steps of:

using a parameterized hash function to map the Java standard symbolic linking strings to linking identifiers, wherein the parameterized hash function uses a parameter stored in a load data structure so that no additional linking information is necessary to translate standard Java class files to Java cap files; and

storing said linking identifiers in an export table.

Claim 25 (currently amended). ~~A~~ The method according to claim 24, wherein a parameterized hash function is used to map standard Java symbolic linking strings onto linking identifiers, wherein each parameter is included in the Java program to be downloaded.

Claim 26 (currently amended). ~~A~~ The method according to claim 25, wherein the parameter for the Java program to be downloaded is used to ensure that the hash function does not map two symbolic linking strings of the Java program to the same linking identifier.

Claim 27 (previously presented). The method of claim 25 wherein the parameter is calculated by a cap file generator.

Claim 28 (previously presented). The method of claim 25 wherein the parameter is calculated by

checking the symbolic linking strings and varying a start parameter until a parameter is formed that satisfies a requirement that the hash function maps all symbolic linking strings on different linking identifiers.

Claim 29 (currently amended). A computer-readable medium, tangibly embodying a program executable by the computer to perform method steps of a method for downloading and linking a Java program on a Java run-time system comprising a stack-based interpreter for executing bytecode instructions, the Java program comprising Java bytecode instructions and Java class structures and including Java standard symbolic linking strings to bind a reference in a bytecode instruction to be executed to a corresponding link target, the method comprising steps of:

using a parameterized hash function for mapping the Java standard symbolic linking strings to linking identifiers, wherein the parameterized hash function uses a parameter stored in a load data structure so that no additional linking information is necessary to translate standard Java class files to Java cap files; and

storing the linking identifiers in an export table.

Claim 30 (previously presented). The medium of claim 29, wherein the export table comprises a Java Card export file.

Claim 31 (previously presented). The medium of claim 29, wherein a parameterized hash function is used to map standard Java symbolic linking strings onto linking identifiers, wherein the parameter is included in the Java program to be downloaded.

Claim 32 (previously presented). The medium of claim 29, wherein the parameter for the Java program to be downloaded is used to ensure that the hash function does not map two symbolic